04-23-07

AFW

ATTORNEY DOCKET NO. 15104.0001U2
EXPRESS MAIL LABEL NO. 915329424 US
PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	)	
	)	
Zygmont et al.	)	Art Unit: **2192**
	)	
Application No. **09/820,185**	)	Examiner: **Fowlkes, Andre R.**
	)	
Filing Date: **March 28, 2001**	)	Confirmation No. **3521**
	)	
For: **"SYSTEM AND METHOD FOR METAPROGRAMMING SOFTWARE DEVELOPMENT ENVIRONMENT**	)	

TRANSMITTAL LETTER

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

NEEDLE & ROSENBERG, P.C.
Customer Number 23859

April 20, 2007

Sir:

Transmitted herewith is the following in the above-identified application:

- ☒ Appeal Brief
- ☐ Fee as calculated below
- ☐ No Additional Fee Required
- ☐ Corrected Drawings
- ☐ Petition to Extend Time
- ☐ Supplemental Declaration
- ☐ Terminal Disclaimer
- ☒ Other Exhibits I - III

	CLAIMS AS AMENDED						
	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE		ADDITIONAL FEE
Total Claims				X $50.00		$0.00
Independent Claims				X $200.00		$0.00
☐ First Presentation of a Multiple Dependent Claim				+ $360.00		$0.00
EXTENSION FEE	1st Month $120 ☐	2nd Month $450 ☐	3rd Month $1020 ☐	4th Month $1590 ☐	5th Month $2160 ☐		$0.00
☐ Reduction by ½ for filing by SMALL ENTITY (Note 37 C.F.R. §1.9, §1.27, §1.28) -							- $0.00
TOTAL FEE DUE							$0.00

553292

Payment:

☐ A check in the amount of $_____ is enclosed.

☐ Payment by credit card in the amount of $0.00 for the fees designated below. (Form PTO-2038 enclosed).
WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

☐ The Commissioner is authorized to charge our Deposit Account No. 14-0629 in the amount of $0.00 to cover the above-listed additional fees. A duplicate copy of this transmittal is enclosed.

☒ In the event of an overpayment or improper payment of a required fee, the Commissioner is authorized to charge or credit our Deposit Account No. 14-0629 as required to correct the error.

Respectfully submitted,

NEEDLE & ROSENBERG, P.C.

Jason S. Jackson
Registration No. 56,733

NEEDLE & ROSENBERG, P.C.
Customer Number 23859
(678) 420-9300
(678) 420-9301 (fax)

CERTIFICATE OF MAILING UNDER 37 C.F.R. § 1.10

I hereby certify that this correspondence, including any items indicated as attached or included, is being deposited with the United States Postal Service as **Express Mail Label No. EV 915329424 US** in an envelope addressed to: **Mail Stop Appeal Brief - Patents**, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date indicated below.

Jason S. Jackson — Date: 4-20-2007

ATTORNEY DOCKET NO. 15104.0001U2
EXPRESS MAIL LABEL NO. EV 915329424 US
PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of	)	
	)	
Zygmont et al.	)	Group Art Unit: **2192**
	)	
Application No. **09/820,185**	)	Examiner: **Fowlkes, Andre R.**
	)	
Filed: **March 28, 2001**	)	Confirmation No. **3521**
	)	
For: **"SYSTEM AND METHOD FOR METAPROGRAMMING SOFTWARE DEVELOPMENT ENVIRONMENT"**	)	

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

NEEDLE & ROSENBERG, P.C.
Customer No. 23859

Sir:

The Appellant submits this brief in connection with the above-identified patent application (hereinafter "Application") and in response to the Notification of Non-Compliant Appeal Brief mailed April 13, 2007. In view of this brief, the Appellant respectfully requests reversal of the rejections and allowance of the pending claims.

(1) REAL PARTY IN INTEREST

The real party in interest is M1 GLOBAL SOLUTIONS, INC., assignee of the Application.

(2) RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences known to Appellant, the undersigned, or Appellant's assignee.

(3) STATUS OF CLAIMS ON APPEAL

Claims 1-40 stand finally rejected by the Examiner. The rejection of claims 1-40 is being appealed.

(4) STATUS OF AMENDMENTS

No amendments have been filed subsequent to the pending final rejection of claims 1-40.

(5) SUMMARY OF THE INVENTION

The independent claims of the Application each provide a novel and unobvious way to develop software by representing computer system architectures as metaprograms, allowing a user of the invention to take a single object model and generate software compatible with any system architecture defined by a metaprogram. Application at page 4, lines 4-17; page 5, lines 15-26; page 10, lines 3-19 and 20-26; page 12, line 13 to page 13, line 3; page 13, line 23 to page 14, line 16.

Independent Claim 1

Claim 1 recites a method for developing a software system, comprising the steps of: providing an object model expressed in an object modeling computer language, the object model representing a software system and comprising components realizing classes (Application at page 4, lines 4-17; page 10, lines 20-26); providing a set of one or more metaprograms reflecting a computer system architecture (Application at page 10, lines 3-19; page 13, line 23 to page 14,

line 16); and a meta-machine binding the components to the metaprograms to generate the software system for a computer system having said architecture (Application at page 5, line 20 to page 6, line 17; page 11, lines 3-5).

Independent Claim 18

Claim 18 recites a meta-development environment computer program product for developing a software system, the computer program product comprising a computer-readable medium carrying thereon: a meta-machine responsive to an object model and a set of one or more metaprograms input to the meta-machine under control of a user (Application at page 5, line 20 to page 6, line 17; page 11, lines 3-5), the object model expressed in an object modeling computer language, the object model representing a software system and comprising components realizing object classes (Application at page 4, lines 4-17; page 10, lines 20-26), the set of metaprograms reflecting a computer system architecture (Application at page 10, lines 3-19; page 13, line 23 to page 14, line 16), the meta-machine under control of the user binding the components to the metaprograms to generate a resultant software system executable on a computer system having the architecture (Application at page 5, line 20 to page 6, line 17; page 11, lines 3-5); and a user interface for providing user control of metaprogram input, object model input, meta-machine operation, and resultant software system output (Application at page 5, lines 20-26; page 11, lines 11-18; page 18, lines 12-15; Figs. 2-3).

Independent Claim 34

Claim 34 recites a method for developing software systems for a plurality of computer system architectures, comprising the steps of: providing a meta-development environment (MDE) to a first party having access to a first computer system with a first architecture (Application at page 5, lines 15-26; page 11, line 11 to page 12, line 7; page 12, line 12 to page 13, line 12; page 14, lines 17-22; Figs. 1-4); providing the MDE to a second party having access to a second computer system with a second architecture (Application at page 5, lines 15-26; page 11, line 11 to page 12, line 7; page 12, line 12 to page 13, line 12; page 14, lines 17-22; Figs. 1-4); providing the first party's MDE with an object model expressed in an object modeling

computer language, the object model representing a software system and comprising components realizing classes (Application at page 4, lines 4-17; page 10, lines 20-26); providing the second party's MDE with the object model (Application at page 4, lines 4-17; page 10, lines 20-26); providing the first party's MDE with a first set of one or more metaprograms reflecting the first architecture, wherein binding in the MDE of the components of the object model and the first set of metaprograms defines first software system code executable on a computer system having the first architecture but not executable on a computer system having the second architecture, the first software system code effecting a user application when executed on the first computer system (Application at page 5, line 20 to page 6, line 17; page 10, lines 3-19; page 11, lines 3-5; page 13, line 23 to page 14, line 16); and providing the second party's MDE with a second set of one or more metaprograms reflecting the second architecture, wherein binding in the MDE of the components of the object model and the second set of metaprograms defines second software system code executable on a computer system having the second architecture but not executable on a computer system having the first architecture, the second software system code effecting the same user application when executed on the second computer system as the first software system code effects when executed on the first computer system (Application at page 5, line 20 to page 6, line 17; page 10, lines 3-19; page 11, lines 3-5; page 13, line 23 to page 14, line 16).

Independent Claim 38

Claim 38 recites a method for developing software systems for a plurality of computer system architectures, comprising the steps of: providing a meta-development environment (MDE) to a first party having access to a first computer system having an architecture (Application at page 5, lines 15-26; page 11, line 11 to page 12, line 7; page 12, line 12 to page 13, line 12; page 14, lines 17-22; Figs. 1-4); providing the MDE to a second party having access to a second computer system having the architecture (Application at page 5, lines 15-26; page 11, line 11 to page 12, line 7; page 12, line 12 to page 13, line 12; page 14, lines 17-22; Figs. 1-4); the first party writing a first object model expressed in an object modeling computer language and loading the first object model into the first party's MDE (Application at page 12, line 12 to page 13, line 12), the object model representing a first software system and comprising

components realizing classes (Application at page 4, lines 4-17; page 10, lines 20-26); the second party writing a second object model expressed in an object modeling computer language and loading the second object model into the second party's MDE (Application at page 12, line 12 to page 13, line 12), the object model representing a second software system and comprising components realizing classes (Application at page 4, lines 4-17; page 10, lines 20-26); the first party purchasing from a vendor a set of one or more metaprograms reflecting the architecture and loading the set of metaprograms into the first party's MDE (Application at page 12, line 12 to page 13, line 12), wherein binding in the MDE of the components of the first object model and the set of metaprograms defines first software system code executable on a computer system having the architecture but not executable on a computer system having another architecture, the first software system code effecting a first user application when executed on the first computer system (Application at page 5, lines 15-26; page 11, line 11 to page 12, line 7; page 12, line 12 to page 13, line 12; page 14, lines 17-22; Figs. 1-4); and the second party purchasing from a vendor a set of one or more metaprograms reflecting the architecture and loading the set of metaprograms into the second party's MDE (Application at page 12, line 12 to page 13, line 12), wherein binding in the MDE of the components of the second object model and the set of metaprograms defines second software system code executable on a computer system having the architecture but not executable on a computer system having another architecture, the second software system code effecting a second user application when executed on the second computer system different from that which the first software system code effects when executed on the first computer system (Application at page 5, lines 15-26; page 11, line 11 to page 12, line 7; page 12, line 12 to page 13, line 12; page 14, lines 17-22; Figs. 1-4).

(6) ISSUES ON APPEAL

Whether claims 1-4, 11-18, 20, 27-35, 37, 38, and 40 are unpatentable under 35 U.S.C. § 102(e) as anticipated by U.S.P.N. 6,715,145 to Bowman-Amuah (hereinafter "Bowman"), and whether claims 5-10, 19, 21-26, 36, and 39 are unpatentable under 35 U.S.C. § 103(a) as being unpatentable over Bowman in view of Mueller, "Instant UML", Wrox Press, 1997 (hereinafter "Mueller").

(7) GROUPING OF CLAIMS

The claims do not stand or fall together. Instead, the Appellant presents separate arguments for various independent and dependent claims. Each of these arguments is separately argued below and is presented with separate headings.

(8) ARGUMENTS

A. Rejection of Claims 1-4, 11-18, 20, 27-35, 37, 38, and 40 under 35 U.S.C. 102(e)

Claims 1-4, 11-18, 20, 27-35, 37, 38, and 40 stand rejected under 35 U.S.C. § 102(e) as anticipated by Bowman in a final Office Action mailed March 2, 2006 (hereinafter "Office Action"). Reversal of these rejections is respectfully requested for at least the reason that the 102 rejection improperly uses a secondary reference to reject each of the independent claims of claims 1, 18, 34, and 38; for at least the reason that the cited references fail to teach or disclose every element of the independent claims; and for at least the reason that the Office Action fails to reject each element of the independent claims.

Independent Claims 1 and 18

The Appellant respectfully requests reversal of the rejections of claims 1 and 18 for at least the reasons that the rejection of claims 1 and 18 improperly relies on a secondary reference; that the cited references fail to disclose every element of claims 1 and 18; and that the Office Action fails to reject every element of claims 1 and 18.

a. Claims 1 and 18 are allowable for at least the reason that the rejection of claims 1 and 18 improperly utilizes a secondary reference.

To support a rejection of claim 1 the Office Action relies on a combination of Bowman and a secondary or "extra" reference. Specifically, the Office Action on page 9 asserts that

Bowman combined with a web page[1] (hereinafter the "HSE Document") discloses:

> -providing a set of one or more metaprograms reflecting a computer system architecture (col. 176:47, "the information model (i.e. a metaprogram: A high-level roadmap containing software, hardware, and other information technology requirements for HSE-MIS, see GEMI's glossary for Health, Safety & Environment – Management Information Systems http://www.hsewebdepot.org/imstool/GEMI.nsf/WEBDocs/Glossary?OpenDocument)),

The Appellant is aware that the use of an "extra" reference in a 102 rejection can be proper in three circumstances as set forth in M.P.E.P. § 2131.01. Specifically, an extra reference can be used to support a primary reference in a 102 rejection to: (1) prove that the primary reference contains an enabling disclosure; (2) explain the meaning of a term used in the primary reference; or (3) show that a characteristic not disclosed in the primary reference is inherent. M.P.E.P. § 2131.01.

Analysis of the pending 102 rejections is hindered because the Office Action does not state why use of the HSE Document is proper under M.P.E.P. § 2131.01. Further, a copy of the HSE Document was not made of record, nor was a publication date for the HSE Document given. In view of these facts the Appellant has considered each potential reason why the HSE Document could be a proper extra reference, and in view of the analysis below, the Appellant can only conclude that the pending 102 rejections are improper under M.P.E.P. § 2131.01.

When a claimed composition or machine is disclosed identically by the primary reference, an additional or "extra" reference may be relied on to show that the primary reference has an enabling disclosure. M.P.E.P. § 2131.01; In re Samour, 571 F.2d 559, 197 USPQ 1 (CCPA 1978); In re Donohue, 766 F.2d 531 (Fed. Cir. 1985). In the Office Action, the Examiner relies on the HSE Document precisely because Bowman does not identically disclose the invention of claim 1. Further, as asserted in the arguments below, neither Bowman nor the HSE Document discloses every element of the independent claims. Thus, Bowman does not identically disclose the invention of the independent claims, such that the HSE Document is not a proper extra reference under M.P.E.P. § 2131.01 if the HSE Document is being used to show that

[1] A copy of the HSE document is included in the Appendix at Exhibit I.

Bowman contains an enabling disclosure.

An extra reference may be used to explain but not expand the meaning of terms and phrases used in the primary reference relied upon as anticipatory of the claimed subject matter. M.P.E.P. § 2131.01; In re Baxter Travenol Labs., 952 F.2d 388 (Fed. Cir. 1991). In the present case, the Examiner alleges equivalence between the "information model" of Bowman and the "information model" of the HSE Document. An analysis of the HSE Document and Bowman shows that the HSE Document is not being used to "explain but not expand" the "information model" of Bowman.

The HSE Document provided by the Office Action exists on a website called the "HSE Web Depot."[2] The "About" page of the HSE Web Depot states:

> About the HSE Web Depot
>
> The GEMI HSE Web Depot, an ongoing collaborative effort of the GEMI IMS Workgroup, is a web-based information resource for Health, Safety & Environment - Management Information Systems (HSE-MIS) development and improvement. The purpose of the tool is to facilitate information sharing among practitioners in this rapidly evolving field. Throughout this tool, we've attempted to minimize the use acronyms. However, please note that within individual case studies we have preserved companies' use of acronyms and terms for their own systems. HSE Web Depot presents a framework for HSE-MIS planning, development, system rollout and improvement and organizes company experiences within these areas in an easy to use format.
>
> . . .
>
> The Global Environmental Management Initiative (GEMI) is a nonprofit organization of leading companies dedicated to fostering environmental, health and safety excellence worldwide through the sharing of tools and information in order for business to help business achieve environmental excellence.[3]

Thus, the HSE Web Depot is directed to using information systems to improve health, safety, and the environment ("HSE"). To this end, the HSE Document discloses an "information model" comprising a high-level roadmap relating to how software, hardware, and information

[2] http://www.hsewebdepot.org, as accessed on January 19, 2007.
[3] http://www.hsewebdepot.org/imstool/GEMI.nsf/WEBDocs/Introduction?OpenDocument, as accessed on January 19, 2007. A copy of this page is included in the Appendix at Exhibit II.

technology can be used to support health, safety, and environmental processes. Specifically, the HSE Document states:

> Information Model: A high-level roadmap containing software, hardware, and other information technology requirements for HSE-MIS. (Emphasis added).

As seen above, the HSE Document equates an "information model" to a "high-level roadmap." The Office Action says nothing about how the "information model" of Bowman could be equivalent or even related to the "high-level roadmap" of the HSE Document. Fortunately, the HSE Web Depot further describes "information model":

> Key Concepts
> * The output of the Information Analysis step is an Information Model.
> * The Information Model defines the data that must be collected to support the HSE processes for today and in the future.
> * The Information Model includes report management, data architecture and standards, information security and the information necessary to support the processes. (Emphasis added).[4]

Thus, the "information model" of the HSE Document is a "high-level roadmap" defining "the data that must be collected to support HSE (Health, Safety, and Environment) processes for today and in the future."

Bowman is directed to structuring batch activities, wherein a series of processing steps is prepared for input objects being streamed into a batch processing system. Bowman at Abstract; Col. 2, lines 19-34. Each of the processing steps of Bowman is encapsulated within a filter, wherein several filters may be used in parallel. Bowman at Col. 2, lines 23-43. Bowman is a voluminous reference, comprising 316 columns and 195 figures, and briefly touches on aspects of software development.

For example, Bowman laments that time is wasted during software development because the output of one development "tool" has to be used as the input for another "tool":

> On a typical project one finds the following tools used in the software development process:

[4] http://www.hsewebdepot.org/imstool/GEMI.nsf/WEBDocs/PlanInformationAnalysis?OpenDocument, as accessed on January 19, 2007. A copy of this page is included in the Appendix at Exhibit III.

> General diagramming tools: Visio, ABC Graphics, etc. for workflow and operation diagrams
>
> MS Office: Word class and component specification templates, Excel scenarios,
>
> Object Oriented CASE tool: class and component models, component/class specifications, message trace diagrams Database design tools: Erwin, Oracle Designer, etc.
>
> Integrated Development Environment(IDE): Visual Studio, Visual Age for Java, JDeveloper, Visual Café
>
> Source code configuration manager: SourceSafe, ClearCase
>
> An inordinate amount of time is invested in the macro process of how to capture and link information in a way that it can be used effectively through the course of the project (e.g., moving the models from the CASE tool into the source code of the targeted IDE environment). Teams should tackle early the selection of deliverables in each phase and which tool the deliverable may be created and maintained within. In addition, they should determine whether the deliverable is to continue to be enhanced in subsequent phases of the project through the iteration process. Bowman at Col. 175, lines 24-55 (Emphasis added).

To remedy the inefficiencies caused by using numerous tools during the development process, Bowman proposes a development architecture that provides seamless integration of development tools:

> Ideally what would greatly increase the productivity of the development architecture is a seamless integration of tools in the workbench and the ability to "plug in" whatever tool is most appropriate for the capture and communication of a particular deliverable. FIG. 50 portrays a development architecture with a seamless integration of tools which can be plugged in for the capture and communication of particular deliverables. Shown in FIG. 50 is the relationship between a process phase 5000, deliverables 5002, tools 5004, repositories 5006, and an information model 5008. Bowman at Col. 175, lines 9-18 (Emphasis added).

Specifically, Bowman proposes integrating software development tools using an "information model":

> A development architecture should provide an environment for component-based solutions that supports a team through the Analysis, Design, and Construction phases of the development process. It should also serve as a productive environment for the on-going maintenance of an application. Conceptually it should integrate all of the necessary tools through an information model and most ideally through a central repository. Bowman at Col. 176, lines 32-29 (Emphasis added).

Thus, Bowman purportedly improves over prior art software development architectures by using an "information model" to facilitate the integration of software development "tools" such as Microsoft Office and Microsoft Visio.

In view of the analysis above, the Appellant asserts that the Examiner is not using the HSE Document to "explain but not expand" the "information model" of Bowman. The "information model" of Bowman is for integrating software development tools in a software development architecture. In contrast, the "information model" of the HSE Document is a "high-level roadmap" defining "data that must be collected" to support health, safety, and environmental processes. The Appellant asserts that the "information model" of Bowman plainly has nothing to do with a "high-level roadmap" for "data that must be collected" to support health, safety, and environmental processes.

In view of these clearly different meanings of the phrase "information model", the Appellant must respectfully assert that the HSE Document is not a proper reference under M.P.E.P. § 2131.01 if the HSE Document is being used to "explain but not expand" the "information model" of Bowman.

An extra reference can be used to show that a characteristic not disclosed in the primary reference is inherent. M.P.E.P. § 2131.01. Further, such an extra reference "must make clear that the missing descriptive matter is necessarily present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill." Continental Can Co. USA v. Monsanto Co., 948 F.2d 1264, 1268 (Fed. Cir. 1991).

In the present case, the Examiner makes no reference to "metaprograms" being a non-disclosed but inherent element of Bowman, but instead asserts that the "information model" of Bowman somehow discloses the metaprograms of claim 1. Further, no showing was made that any missing element of Bowman was necessarily present in Bowman and would have been so recognized by persons of ordinary skill in the art. Thus, the HSE Document is not a proper reference under M.P.E.P. § 2131.01 if the HSE document is being used to show that metaprograms as recited in claim 1 are inherently disclosed by Bowman.

In view of the analysis above, the Appellant respectfully asserts that the Examiner's use of the HSE Document as an extra reference fails to satisfy any of the three rules provided by M.P.E.P. § 2131.01. Accordingly, claim 1 is allowable for at least the reason that the Examiner's use of the HSE Document creates an improper 102 rejection.

b. Claims 1 and 18 are allowable for at least the reason that the cited references do not teach or disclose every element of claims 1 and 18, and for at least the reason that the Office Action fails to reject every element of claims 1 and 18.

Claim 1 stands rejected as anticipated by Bowman. A proper rejection of a claim under 35 U.S.C. § 102 requires that a single prior art reference disclose each element of the claim. W.L. Gore & Assoc., Inc., v. Garlock, Inc., 721 F.2d 1540 (Fed. Cir. 1983). Anticipation requires that each and every element of the claimed invention be disclosed in a single prior art reference. In re Paulsen, 30 F.3d 1475 (Fed. Cir. 1994). For anticipation, there must be no difference between the claimed invention and the reference disclosure as viewed by a person of ordinary skill in the field of the invention. Scripps Clinic & Res. Found. v. Genentech. Inc., 927 F.2d 1565, 18 (Fed. Cir. 1991).

Claim 1 of the Application recites a novel and unobvious method for developing a software system by binding components expressed in an object modeling computer language to one or more metaprograms reflecting a computer system architecture, resulting in the generation of software that is compatible with the architecture reflected by the one or more metaprograms.

Prior art object modeling systems can produce software from a given object model. However, prior art object modeling systems only generate software for specific computer system architectures. For example, a prior art object modeling system could take an object model and generate software for a specific architecture, such as a computer operating Microsoft Windows XP and an Oracle database. That same object modeling system, however, could not use that object model to generate software that would operate on a different architecture, such as a computer system operating Linux and an Oracle database.

The invention of claim 1 distinguishes over prior art object modeling systems by abstracting out system architectures into one or more metaprograms, enabling, for example, a single object model to be used for generating software for any architecture that is defined by a metaprogram. Thus, a hypothetical developer using the invention of claim 1 can focus his or her time and effort on making object models, because, once complete, the developer can use those object models to generate software for any architecture defined by a metaprogram.

As discussed above, claim 1 recites a method for developing software utilizing metaprograms that reflect a computer system architecture, and includes the step of providing a set of one or more metaprograms reflecting a computer system architecture.

To support a rejection of claim 1 as anticipated by Bowman, the Examiner asserts on page 9 of the Office Action that the cited references disclose:

> -providing a set of one or more metaprograms reflecting a computer system architecture (col. 176:47, "the information model (i.e. a metaprogram: A high-level roadmap containing software, hardware, and other information technology requirements for HSE-MIS, see GEMI's glossary for Health, Safety & Environment – Management Information Systems http://www.hsewebdepot.org/imstool/GEMI.nsf/WEBDocs/Glossary?OpenDocument)),

The Appellant strongly disagrees with the Examiner's interpretation of claim 1 and characterization of the references. First, as discussed above, the "information model" of Bowman is used to integrate software development tools such as Microsoft Office and Microsoft Visio in a software development architecture. Integrating Microsoft Office into a software development architecture does not disclose, and simply is not relevant to, a metaprogram

reflecting a computer system architecture as recited in claim 1.

Second, as discussed above, an "information model" as disclosed by the HSE Document is a "high-level roadmap" defining "data that must be collected" to support health, safety, and environmental processes. Such a "high-level roadmap" does not disclose, and clearly is not relevant to, a metaprogram reflecting a computer system architecture as recited in claim 1.

In the first case, the Examiner uses a single phrase from Bowman (information model), taken dramatically out of context, to reject claim 1. The Examiner then attempts to mitigate this clear deficiency by asserting in the second case that the "information model" of Bowman and the "information model" of the HSE Document are equivalent and therefore, substitutable. Such a substitution is improper on its face because of the two clearly different meanings of "information model" provided by Bowman and the HSE Document, as discussed above. Further, assuming arguendo that the Examiner's statement is true, the rejection is improper because it assigns two clearly different meanings to the metaprogram of claim 1.

Accordingly, claim 1 is allowable for at least the reason that neither of the cited references disclose providing a set of one or more metaprograms reflecting a computer system architecture as recited in claim 1.

As discussed above, claim 1 generates software for a particular architecture by binding components to one or more metaprograms reflecting that architecture. To this end, claim 1 recites "a meta-machine binding the components to the metaprograms to generate the software system for a computer system having said architecture."

To support a rejection of claim 1, the Office Action asserts on page 10 that Bowman discloses:

> - a meta-machine binding the components to the metaprograms to generate the software system for a computer system having said architecture (col. 176:37-38, "it should integrate (i.e. bind) all of the necessary tools (i.e. components) through an information model")

The Appellant strongly disagrees with the Examiner's interpretation of claim 1 and characterization of the references. First, assuming arguendo that the Examiner's characterization of Bowman is accurate, claim 1 is still allowable. Specifically, performing the substitutions

provided by the Examiner (and assuming arguendo that the "information model" of Bowman discloses a metaprogram as recited in claim 1) yields something like the phrase "it should bind all of the components through a metaprogram." The resulting phrase is simply nonsensical because components are not bound "through" a "metaprogram"; rather, components are binded to or with metaprograms to generate software.

Further, the resulting phrase fails to disclose or even address the generation of software for a computer system having the architecture reflected by the one or more metaprograms as recited in claim 1. Thus, assuming arguendo that the Examiner's characterization of Bowman, above, is accurate, claim 1 is allowable for at least the reason that the Office Action fails to reject every element of claim 1.

Second, the rejection is improper because the Examiner inconsistently interprets the "components" of claim 1. The Examiner asserts on page 9 of the Office Action that the phrase "object model representing a software system and comprising components realizing classes" as recited in the providing step of claim 1 is disclosed by Bowman at Col. 176, line 44, stating on page 9 of the Office Action that:

> "the Unified Modeling Language (UML)", and UML provides the capability to express object models representing software system and comprising components realizing classes) (Emphasis added).

The Examiner then asserts on page 10 of the Office Action that Bowman discloses:

> - a meta-machine binding the components to the metaprograms to generate the software system for a computer system having said architecture (col. 176:37-38, "it should integrate (i.e. bind) all of the necessary tools (i.e. components) through an information model") (Emphasis added).

A reading of claim 1 clearly shows that the "components" of the providing step are the same "components" used by a meta-machine to generate software as recited in claim 1. Yet, the Examiner assigns two different meanings to "components": first, the Examiner asserts that "components" are anticipated by UML; then, the Examiner asserts that "components" are anticipated by "tools" as disclosed by Bowman. As discussed above, the "tools" of Bowman are software packages such as Microsoft Office and Microsoft Visio. It is readily apparent to one of skill in the art that "components realizing classes" are not "tools" like Microsoft Visio. Thus, the

rejection of claim 1 is improper for at least the reason that the Office Action assigns two different meanings to the "components" of claim 1.

Third, putting this inconsistency aside, claim 1 is still allowable because the "tools" of Bowman do not teach or disclose components as recited in claim 1. Claim 1 has nothing to do with "tools," but recites that an object model is expressed in an object modeling computer language, and that object models comprise components realizing classes. The Appellant asserts that components as recited in the claim 1 are clearly not disclosed by a "tool" like Microsoft Visio. Claim 1 is thus allowable for at least this reason.

Fourth, claim 1 is allowable for at least the reason that "integrate" and "bind" are not equivalent as asserted by the Examiner. As understood by one of skill in the art, things such as components, classes, and variables are "binded" to generate software; an assertion that "tools" such as Microsoft Office and Microsoft Visio are "binded" is nonsensical. Thus, claim 1 is allowable for at least the reason that Bowman does not disclose binding components to metaprograms as recited in claim 1.

Claim 18 recites a program product that includes the elements of claim 1. Accordingly, the Appellant asserts that claim 18 is allowable for at least the reasons given for the allowability of claim 1.

Dependent Claim 3

Claim 3 is allowable for at least the reason that the Office Action fails to reject every element of claim 3. Specifically, the Office Action on page 10 addresses "the step of providing an object model", but does not address that "the step of providing an object model is performed in part by a business analyst creating the object model" as recited in claim 3.

Similarly, the Office Action addresses "the step of providing a set of one or more metaprograms", but does not address that "the step of providing a set of one or more metaprograms is performed in part by a technologist coding the metaprograms" as recited in claim 3. Claim 3 is also allowable for at least the reason that it depends from allowable claim 1.

Dependent Claims 12 and 28

Claim 12 is allowable for at least the reason that passing references to "Code generation" and "code generation tools" as provided on page 11 of the Office Action do not and simply cannot teach or disclose a metaprogram including code and metacode as recited in claim 12. Claim 12 is also allowable for at least the reason that it depends from allowable claim 1.

Claim 28 recites elements of claim 12, and so is allowable for at least the reasons given for the allowability of claim 12. Claim 28 is also allowable for at least the reason that it depends from allowable claim 18.

Dependent Claims 15 and 31

Claim 15 is allowable for at least the reason that passing references to "Code generation" and "code generation tools" as provided on page 12 of the Office Action do not and simply cannot teach or disclose that the set of metaprograms includes a model metaprogram that modifies the object model as recited in claim 15. Claim 15 is also allowable for at least the reason that it depends from allowable claim 1.

Further, it is clearly inconsistent to assert that "Code generation" and "code generation tools" disclose both that a metaprogram includes code and metacode as recited in claim 12 and that the set of metaprograms includes a model metaprogram that modifies the object model as recited in claim 15.

Claim 31 recites elements of claim 15, and so is allowable for at least the reasons given for the allowability of claim 15. Claim 31 is also allowable for at least the reason that it depends from allowable claim 18.

Dependent Claims 16 and 32

Claim 16 is allowable for at least the reason that passing references to "Code generation" and "code generation tools" as provided in the Office Action on page 12 do not and simply cannot teach or disclose that the set of metaprograms includes a component metaprogram invoked once for each component and uses the classes realized by the component to produce a portion of the software system as recited in claim 16. Claim 16 is also allowable for at least the

reason that it depends from allowable claim 1.

Further, it is clearly inconsistent to assert that "Code generation" and "code generation tools" disclose both that a metaprogram includes code and metacode as recited in claim 12 and that the set of metaprograms includes a component metaprogram invoked once for each component and uses the classes realized by the component to produce a portion of the software system as recited in claim 16.

Claim 32 recites elements of claim 16, and so is allowable for at least the reasons given for the allowability of claim 16. Claim 32 is also allowable for at least the reason that it depends from allowable claim 18.

Dependent Claims 17 and 33

Claim 17 is allowable for at least the reason that passing references to "Code generation" and "code generation tools" as provided in the Office Action at page 13 do not and simply cannot to teach or disclose that the set of metaprograms includes a class metaprogram that is invoked once for each class realized in each component and that produces a portion of the software system as recited in claim 17.

Further, it is clearly inconsistent to assert that "Code generation" and "code generation tools" disclose both that a metaprogram includes code and metacode as recited in claim 12 and that the set of metaprograms includes a class metaprogram that is invoked once for each class realized in each component and that produces a portion of the software system as recited in claim 17.

Claim 33 recites elements of claim 17, and so is allowable for at least the reasons given for the allowability of claim 17. Claim 33 is also allowable for at least the reason that it depends from allowable claim 18.

Dependent Claims 4 and 20

Claim 4 is allowable for at least the reason that a passing reference to a "GUI" as provided on page 10 of the Office Action does not and simply cannot teach or disclose that providing one or more metaprograms comprises a user using a graphical user interface to list one

or more metaprojects in a second window as recited in claim 4.

Claim 20 recites elements of claim 4, and so is allowable for at least the reasons given for the allowability of claim 4. Claim 20 is also allowable for at least the reason that it depends from allowable claim 18.

Dependent Claims 13, 27, and 29

Claim 13 is allowable for at least the reason that a passing reference to a "GUI" on page 11 of the Office Action does not and simply cannot to teach or disclose a user using a graphical user interface to invoke a metaprogram editor as recited in claim 13. It is also clearly inconsistent to assert that a superficial reference to a "GUI" discloses both claim 4 and claim 13, which recite different subject matter.

Claim 27 recites elements from claim 13, and so is allowable for at least the reasons given for the allowability of claim 13. Claim 29 also recites elements of claim 13, and so is allowable for at least the reasons given for the allowability of claim 13. Claims 13, 27, and 29 are also allowable for at least the reason that each depends from an allowable claim.

Dependent Claims 14 and 30

Claim 14 is allowable for at least the reason that the passing reference to a "GUI" on page 12 of the Office Action does not and simply cannot teach or disclose a user activating a toggling function of the metaprogram editor to toggle a window between highlighting the code and highlighting the metacode as recited in claim 14. It is also clearly inconsistent to assert that a superficial reference to a "GUI" can disclose each of claims 4, 13, and 14, which recite different subject matter.

Claim 30 recites elements of claim 14, and so is allowable for at least the reasons given for the allowability of claim 14. Claim 30 is also allowable for at least the reason that it depends from allowable claim 18.

Independent Claim 34

Claim 34 recites a method for developing software systems for a plurality of computer system architectures and recites several elements of claim 1, such as binding components to metaprograms to generate software. Thus, claim 34 is allowable for at least the reasons given for the allowability of claim 1.

To support a rejection of claim 34, the Office Action asserts on page 13 that:

> As per claims 20, 27-35, 37, 38, and 40, Bowman also discloses such claimed limitations as addressed in claims 3, 4 and 12-17.

The Appellant disagrees with the above statement, and asserts that claim 34 contains several elements that are not recited by claims 3, 4, or 12-17. For example, claim 34 recites the steps of:

> providing a meta-development environment (MDE) to a first party having access to a first computer system with a first architecture;
>
> providing the MDE to a second party having access to a second computer system with a second architecture;
>
> providing the first party's MDE with an object model expressed in an object modeling computer language, the object model representing a software system and comprising components realizing classes;
>
> providing the second party's MDE with the object model;
>
> providing the first party's MDE with a first set of one or more metaprograms reflecting the first architecture, wherein binding in the MDE of the components of the object model and the first set of metaprograms defines first software system code executable on a computer system having the first architecture but not executable on a computer system having the second architecture, the first software system code effecting a user application when executed on the first computer system;

None of claims 3, 4, or 12-17 recite the steps presented above for at least the reason that claims 3, 4, or 12-17 do not recite a first party and a second party as recited in claim 34. Accordingly, the Appellant asserts that claim 34 is allowable for at least the reason that the Office Action fails to reject every step of claim 34, as is required for a proper 102 rejection.

Independent Claim 38

Claim 38 recites a method for developing software systems for a plurality of computer system architectures, and recites several elements of claim 34. Thus, claim 38 is allowable for at least the reasons given for the allowability of claim 34.

Claim 38 also recites several elements that are not recited by claims 3, 4, 12-17, or 34, such as:

> the first party purchasing from a vendor a set of one or more metaprograms reflecting the architecture and loading the set of metaprograms into the first party's MDE, wherein binding in the MDE of the components of the first object model and the set of metaprograms defines first software system code executable on a computer system having the architecture but not executable on a computer system having another architecture, the first software system code effecting a first user application when executed on the first computer system; and
>
> the second party purchasing from a vendor a set of one or more metaprograms reflecting the architecture and loading the set of metaprograms into the second party's MDE, wherein binding in the MDE of the components of the second object model and the set of metaprograms defines second software system code executable on a computer system having the architecture but not executable on a computer system having another architecture, the second software system code effecting a second user application when executed on the second computer system different from that which the first software system code effects when executed on the first computer system.

The Office Action simply does not address or reject at least the two steps above. Thus, claim 38 is allowable for at least the reason that the Office Action fails to reject every step of claim 38, as is required for a proper 102 rejection.

Dependent Claims 35, 37, and 40

The Appellant respectfully asserts that claims 35, 37, and 40 are allowable for at least the reason that each depends directly or indirectly from allowable claim 34 or 38.

B. Rejection of Claims 5-10, 19, 21-26, 36, and 39 under 35 U.S.C. 103(a)

Claims 5-10, 19, 21-26, 36, and 39 are rejected as obvious over Bowman in view of Mueller. Reversal of these rejections is respectfully requested for at least the reason that the cited references fail to disclose every element of claims 5-10, 19, 21-26, 36, and 39 as is required for an obviousness rejection.

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicants disclosure. In re Vaeck, 947 F.2d 488 (Fed. Cir. 1991).

Dependent Claim 8

Claim 8 is allowable for at least the reason that a passing reference to "Code generation" as provided on page 16 of the Office Action does not and simply cannot teach or disclose searching the list of metaprojects for a metaproject having a name matching a name of an implementation target as recited in claim 8.

Similarly, claim 8 is allowable for at least the reason that the same passing reference to "Code generation" fails to disclose storing an indication of an association between the metaprojects having the matching name with metaprograms of which representations are listed in the metaprojects having the matching name as recited in claim 8.

Further, it is clearly inconsistent to assert that "Code generation" discloses two plainly different elements of the same claim; namely that it discloses both "searching the list of metaprojects" and "storing an indication of an association between the metaprojects", each as recited in claim 8.

Claims 5-10, 19, 21-26, 36, and 39

As discussed above, the cited references fail to teach or disclose every element of the independent claims of claims 1, 18, 34, and 38, from which claims 5-10, 19, 21-26, 36, and 39 depend. Thus, claims 5-10, 19, 21-26, 36, and 39 are allowable for at least the reason that cited references fail to disclose every element of claims 5-10, 19, 21-26, 36, and 39 as is required to establish a *prima facie* case of obviousness.

CONCLUSION

For at least the reasons above, the pending claims are believed to be patentable over the cited references. Accordingly, the Appellant requests reversal of the rejections and allowance of the pending claims. The Commissioner is hereby authorized to charge any additional fees which may be required, or credit any overpayment to Deposit Account No. 14-0629.

Respectfully Submitted,

NEEDLE & ROSENBERG, P.C.

Jason S. Jackson
Registration No. 56,733

Customer No. 23859
Tel: 678-420-9300
Fax: 678-420-9301

CERTIFICATE OF MAILING UNDER 37 C.F.R. § 1.10

I hereby certify that this correspondence, including any items indicated as attached or included, is being deposited with the United States Postal Service as **Express Mail Label No. EV 915329424 US** in an envelope addressed to: **Mail Stop Appeal Brief - Patents**, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date indicated below.

Jason S. Jackson

4-20-2007
Date

(9) CLAIMS APPENDIX

1. A method for developing a software system, comprising the steps of:

providing an object model expressed in an object modeling computer language, the object model representing a software system and comprising components realizing classes;

providing a set of one or more metaprograms reflecting a computer system architecture; and

a meta-machine binding the components to the metaprograms to generate the software system for a computer system having said architecture.

2. The method claimed in claim 1, wherein the object modeling computer language is the Unified Modeling Language (UML).

3. The method claimed in claim 1, wherein:

the step of providing an object model is performed in part by a business analyst creating the object model; and

the step of providing a set of one or more metaprograms is performed in part by a technologist coding the metaprograms.

4. The method claimed in claim 1, wherein:

the step of providing an object model comprises a user using a graphical user interface to list a project in a first window, the project representing the object model; and

the step of providing one or more metaprograms comprises a user using a graphical user interface to list one or more metaprojects in a second window, each metaproject including a list of representations of the metaprograms.

5. The method claimed in claim 4, wherein the object modeling computer language includes an extension mechanism for the specification of user-defined properties and the assignment of

these properties and their values to elements of the object model.

6. The method claimed in claim 5, wherein:
the object modeling computer language is the Unified Modeling Language (UML); and
the user-defined extension mechanism is a Stereootype.

7. The method claimed in claim 5, wherein:
the object modeling computer language is the Unified Modeling Language (UML); and
the user-defined extension mechanism is Tagged Values.

8. The method claimed in claim 5, wherein the step of a meta-machine binding the components to the metaprograms comprises the steps of:
searching the list of metaprojects for a metaproject having a name matching a name of an implementation target, wherein the implementation target is defined by the user-defined extension mechanism associated with the components; and
storing an indication of an association between the metaproject having the matching name with metaprograms of which representations are listed in the metaproject having the matching name.

9. The method claimed in claim 8, wherein:
the object modeling computer language is the Unified Modeling Language (UML);
the user-defined extension mechanism is a Stereootype; and
the Stereotype indicates the implementation target.

10. The method claimed in claim 8, wherein:
the object modeling computer language is the Unified Modeling Language (UML);
the user-defined extension mechanism is Tagged Values; and
the Tagged Values indicate the implementation target.

11. The method claimed in claim 1, further comprising the step of a user using a graphical

user interface to invoke a metaprogram editor.

12. The method claimed in claim 1, wherein each metaprogram in said set of metaprograms includes code and metacode, and the metacode generates a portion of the source code of the software system by outputting the code.

13. The method claimed in claim 12, further comprising the step of a user using a graphical user interface to invoke a metaprogram editor.

14. The method claimed in claim 13, further comprising the step of a user activating a toggling function of the metaprogram editor to toggle a window between highlighting the code and highlighting the metacode.

15. The method claimed in claim 1, wherein the set of metaprograms includes a model metaprogram that modifies the object model.

16. The method claimed in claim 1, wherein the set of metaprograms includes a component metaprogram invoked once for each component and uses the classes realized by the component to produce a portion of the software system.

17. The method claimed in claim 1, wherein the set of metaprograms includes a class metaprogram that is invoked once for each class realized in each component and that produces a portion of the software system.

18. A meta-development environment computer program product for developing a software system, the computer program product comprising a computer-readable medium carrying thereon:
a meta-machine responsive to an object model and a set of one or more metaprograms input to the meta-machine under control of a user, the object model expressed in an object modeling

computer language, the object model representing a software system and comprising components realizing object classes, the set of metaprograms reflecting a computer system architecture, the meta-machine under control of the user binding the components to the metaprograms to generate a resultant software system executable on a computer system having the architecture; and
a user interface for providing user control of metaprogram input, object model input, meta-machine operation, and resultant software system output.

19. The computer program product claimed in claim 18, wherein the object modeling computer language is the Unified Modeling Language (UML).

20. The computer program product claimed in claim 18, wherein the user interface comprises a graphical user interface listing a project in a first window, the project representing the object model, and listing one or more metaprojects in a second window, each metaproject including a list of representations of the metaprograms.

21. The computer program product claimed in claim 20, wherein the object modeling computer language includes an extension mechanism for the specification of user-defined properties and the assignment of these properties and their values to elements of the object model.

22. The computer program product claimed in claim 21, wherein:
the object modeling computer language is the Unified Modeling Language (UML); and
the user-defined extension mechanism is a Stereootype.

23. The computer program product claimed in claim 21, wherein:
the object modeling computer language is the Unified Modeling Language (UML); and
the user-defined extension mechanism is Tagged Values.

24. The computer program product claimed in claim 21, wherein the meta-machine binds the

components to the metaprograms by searching the list of metaprojects for a metaproject having a name matching a name of an implementation target and storing an indication of an association between the metaproject having the matching name with metaprograms of which representations are listed in the metaproject having the matching name, wherein the implementation target is defined by the user-defined extension mechanism associated with the components.

25. The computer program product claimed in claim 24, wherein:
the object modeling computer language is the Unified Modeling Language (UML);
the user-defined extension mechanism is a Stereotype; and
the Stereotype indicates the implementation target.

26. The computer program product claimed in claim 24, wherein:
the object modeling computer language is the Unified Modeling Language (UML);
the user-defined extension mechanism is Tagged Values; and
the Tagged Values indicate the implementation target.

27. The computer program product claimed in claim 18, wherein the user interface includes a metaprogram editor.

28. The computer program product claimed in claim 18, wherein each metaprogram in said set of metaprograms includes code and metacode, and the metacode generates a portion of the source code of the software system by outputting the code.

29. The computer program product claimed in claim 28, wherein the user interface includes a metaprogram editor.

30. The computer program product claimed in claim 29, wherein the metaprogram editor includes a toggling means for toggling a window between highlighting the code and highlighting the metacode.

31. The computer program product claimed in claim 18, wherein the set of metaprograms includes a model metaprogram that modifies the object model.

32. The computer program product claimed in claim 18, wherein the set of metaprograms includes a component metaprogram invoked once for each component and uses the classes realized by the component to produce a portion of the software system.

33. The computer program product claimed in claim 18, wherein the set of metaprograms includes a class metaprogram that is invoked once for each class realized in each component and that produces a portion of the software system.

34. A method for developing software systems for a plurality of computer system architectures, comprising the steps of:

providing a meta-development environment (MDE) to a first party having access to a first computer system with a first architecture;

providing the MDE to a second party having access to a second computer system with a second architecture;

providing the first party's MDE with an object model expressed in an object modeling computer language, the object model representing a software system and comprising components realizing classes;

providing the second party's MDE with the object model;

providing the first party's MDE with a first set of one or more metaprograms reflecting the first architecture, wherein binding in the MDE of the components of the object model and the first set of metaprograms defines first software system code executable on a computer system having the first architecture but not executable on a computer system having the second architecture, the first software system code effecting a user application when executed on the first computer system; and

providing the second party's MDE with a second set of one or more metaprograms

reflecting the second architecture, wherein binding in the MDE of the components of the object model and the second set of metaprograms defines second software system code executable on a computer system having the second architecture but not executable on a computer system having the first architecture, the second software system code effecting the same user application when executed on the second computer system as the first software system code effects when executed on the first computer system.

35. The method claimed in claim 34, wherein:

the MDE includes a metaprogram editor and a meta-machine;

the step of providing the first party's MDE with a first set of one or more metaprograms includes the step of the first party using the metaprogram editor of the first party's MDE to write metaprogram code; and

the step of providing the second party's MDE with a second set of one or more metaprograms includes the step of the second party using the metaprogram editor of the second party's MDE to write metaprogram code.

36. The method claimed in claim 35, wherein the object modeling computer language is the Unified Modeling Language (UML).

37. The method claimed in claim 35, wherein:

the step of providing the first party's MDE with an object model comprises the first party purchasing the object model from a vendor and loading the object model into the first party's MDE; and

the step of providing the second party's MDE with an object model comprises the second party purchasing the same object model from a vendor and loading the object model into the second party's MDE.

38. A method for developing software systems for a plurality of computer system architectures, comprising the steps of:

providing a meta-development environment (MDE) to a first party having access to a first computer system having an architecture;

providing the MDE to a second party having access to a second computer system having the architecture;

the first party writing a first object model expressed in an object modeling computer language and loading the first object model into the first party's MDE, the object model representing a first software system and comprising components realizing classes;

the second party writing a second object model expressed in an object modeling computer language and loading the second object model into the second party's MDE, the object model representing a second software system and comprising components realizing classes;

the first party purchasing from a vendor a set of one or more metaprograms reflecting the architecture and loading the set of metaprograms into the first party's MDE, wherein binding in the MDE of the components of the first object model and the set of metaprograms defines first software system code executable on a computer system having the architecture but not executable on a computer system having another architecture, the first software system code effecting a first user application when executed on the first computer system; and

the second party purchasing from a vendor a set of one or more metaprograms reflecting the architecture and loading the set of metaprograms into the second party's MDE, wherein binding in the MDE of the components of the second object model and the set of metaprograms defines second software system code executable on a computer system having the architecture but not executable on a computer system having another architecture, the second software system code effecting a second user application when executed on the second computer system different from that which the first software system code effects when executed on the first computer system.

39. The method claimed in claim 38, wherein the object modeling computer language is the Unified Modeling Language (UML).

40. The method claimed in claim 38, wherein:

the step of the first party writing a first object model comprises a business analyst writing the first object model; and

the step of the second party writing a second object model comprises a business analyst writing the second object model.

(10) EVIDENCE APPENDIX

Exhibit I The HSE Document at http://www.hsewebdepot.org/imstool/GEMI.nsf/WEBDocs/Glossary?OpenDocument, as accessed on June 3, 2006.

Exhibit II The "About" page from the HSE Web Depot at http://www.hsewebdepot.org/imstool/GEMI.nsf/WEBDocs/Introduction?OpenDocument, as accessed on January 19, 2007.

Exhibit III The page from the HSE Web Depot describing "Key Concepts" at http://www.hsewebdepot.org/imstool/GEMI.nsf/WEBDocs/PlanInformationAnalysis?OpenDocument, as accessed on January 19, 2007.

(11) RELATED PROCEEDINGS INDEX

None.

HSE Web Depot

Glossary

Home Page

Cross-functional team: Group of expert representatives from all relevant business units to guide the Plan Phase. Cross-functional teams typically include: HSE, MIS, Manufacturing and Operations, Marketing, Research and Development, Finance, Purchasing, Public Affairs and Investor Relations, and Legal. All are critical stakeholders in the HSE-MIS system.

Current ("as-is") state: A description of the current situation in terms of the work processes, HSE-information and MIS performance.

Data architecture: The overall software configuration and data linkage supporting a particular HSE-MIS application.

Enterprise Resource Planning (ERP): An industry term for the broad set of activities supported by application software that helps a manufacturer or other business manage the important parts of its business, including product planning, parts purchasing, maintaining inventories, interacting with suppliers, providing customer service, and tracking orders. ERP systems provide an opportunity to support the integration of HSE activities into the business.

Future state ("to-be"): A description of the desired future state of the redesigned or newly design HSE-MIS requirements for the reengineered work process.

Globalization: A set of processes leading to the integration of economic, cultural, political, and social systems across geographical boundaries.

HSE: Health, Safety and Environment.

HSE-MIS: Health, Safety and Environment- Management Information Systems - a set of processes and systems supporting the HSE functions.

HSE Business Plan: The HSE plan that identifies the goals, objectives and targets to achieve the HSE vision, mission, policy and business requirements.

HSE-MIS Development Initiative: Description of a need identified by the gap analysis.

EXHIBIT I

Strategic Business Plan: A strategic roadmap that competitively differentiates a company's products and services, target markets, customers, the competition and the resources needed to deliver them successfully to customers.

Sustainability: There are many definitions of sustainability, but a good one for the HSE-MIS context is:

> "Sustainable means using methods, systems and materials that won't deplete resources or harm natural cycles" (Rosenbaum, 1993).

System Architecture: The overall hardware/software configuration and database design supporting a particular HSE-MIS application, including internet, intranet and extranet network linkages.

System Development Life Cycle: Series of steps required to conceptualize, design, and implement a project such as: Analysis, Design, Building, Implementation, Operation and Maintenance.

Test Scripts: Specific application simulations to beta test a new or modified software application.

Testing Scenarios: Different application circumstances and conditions that validate the software's capabilities to meet the functional requirements.

HSE Software Product: A commercially developed software product designed to support a specific HSE application or group of HSE applications.

Information Model: A high-level roadmap containing software, hardware, and other information technology requirements for HSE-MIS.

Implementation Plan: Comprehensive outline of the most appropriate options for implementing initiatives identified and ranked during the Gap Analysis and Prioritization steps.

Implementation Scenario: A detailed view of what is required to implement a specific initiative.

Integrator: Individual or group that ensures that the HSE systems and processes that are developed and implemented fit into overall business functions.

Internal Rate of Return: The rate of discount that makes the net present value (NPV) equal to zero.

MIS: Management Information System(s).

Net Present Value: The present value of future cash returns of a project.

Owners/Ownership: Business unit/function responsible for collecting, using, reporting and maintaining the HSE-MIS information.

Prototypes: Smaller versions of the system that can be validated quickly by the user without construction of an entire system.

Return on Investment: Benefits of a project (over a set time period) divided by the amount invested in the project.

Screen Shots: Graphic representations of screens the user would see on the finished system. They are built with limited functionality behind the screens and are intended to show high-level functionality with a detailed user interface.

Source Code: Commercially or in-house developed proprietary algorithms, macros, etc. that define how the software program works, completes its computations, etc.

Sponsors/Sponsorship: Business unit /function providing the business resources to support the HSE-MIS application.

HSE Web Depot

Introduction

Home Page

About the HSE Web Depot

The GEMI HSE Web Depot, an ongoing collaborative effort of the GEMI IMS Workgroup, is a web-based information resource for Health, Safety & Environment - Management Information Systems (HSE-MIS) development and improvement. The purpose of the tool is to facilitate information sharing among practitioners in this rapidly evolving field. Throughout this tool, we've attempted to minimize the use acronyms. However, please note that within individual case studies we have preserved companies' use of acronyms and terms for their own systems. HSE Web Depot presents a framework for HSE-MIS planning, development, system rollout and improvement and organizes company experiences within these areas in an easy to use format.

It contains information on:

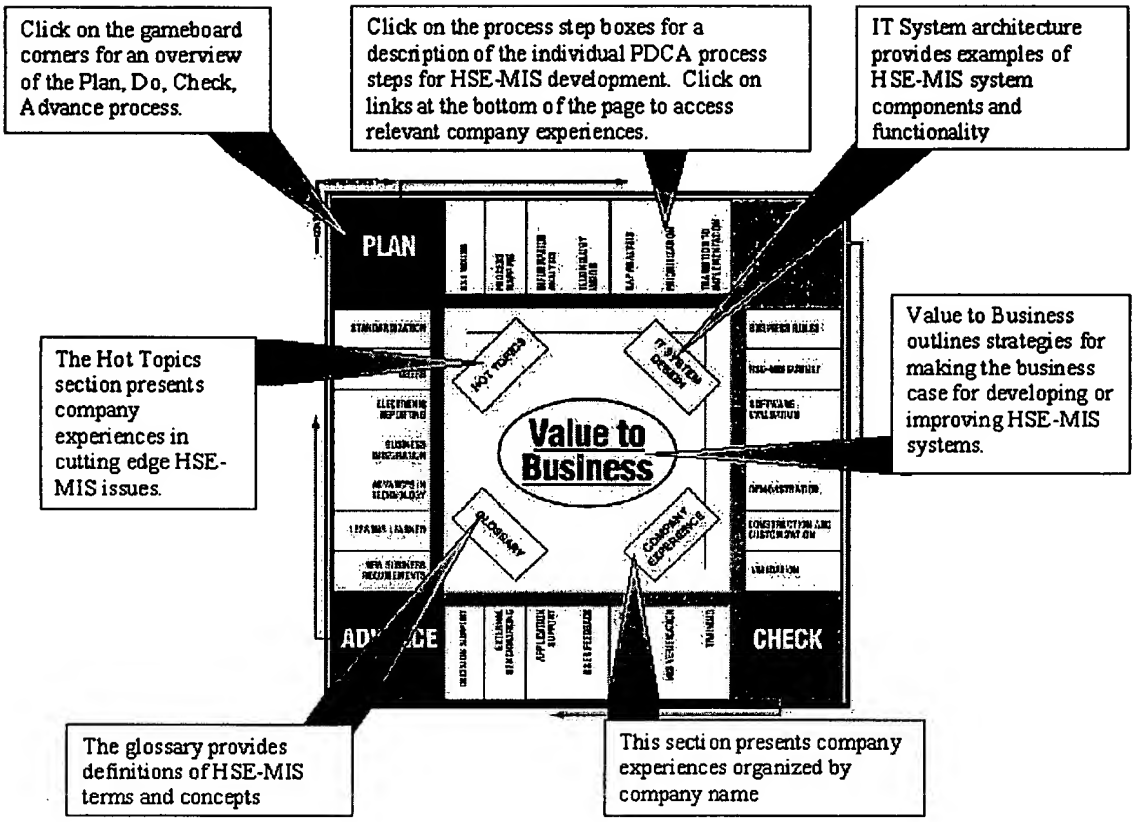
- Business benefits and efficiencies of HSE-MIS;
- Best practices of world class companies in the field of HSE-MIS;
- Strategies for planning, designing, evaluating, and improving HSE-MIS; and
- Examples of HSE-MIS architecture and functionality.

HSE Web Depot is a living, growing document, and new content and company experiences will continually be added to allow users to keep pace with new developments, challenges, and opportunities. For more information about the tool, or if you're interested in sharing your company's experiences, please contact GEMI at gemi@worldweb.net or 202-296-7449.

Start Using HSE Web Depot

How to Use the HSE Web Depot

EXHIBIT II



Helpful hint: use the F11 key to toggle between normal and full screen views to fit the entire gameboard on your computer screen.

Start Using HSE Web Depot

The Global Environmental Management Initiative (GEMI) is a nonprofit organization of leading companies dedicated to fostering environmental, health and safety excellence worldwide through the sharing of tools and information in order for business to help business achieve environmental excellence.

The guidance included in this web site is based on the professional judgment of the individual collaborators listed in the Site Credits. Site content does not necessarily represent the opinion of GEMI or of any GEMI member company or consultant. Responsibility for any application of the guidance in this website rests solely with the user.

Plan - Information Analysis

HSE Information Analysis involves defining data requirements and architecture, standards and measures, and report delivery and security. Information Analysis identifies the information and data needed to support HSE processes, reporting and decision-making today and in the future.

Key Concepts

- The output of the Information Analysis step is an Information Model.
- The Information Model defines the data that must be collected to support the HSE processes for today and in the future.
- The Information Model includes report management, data architecture and standards, information security and the information necessary to support the processes.

Practical Advice

- Once a company has gone through the planning cycle, a current HSE Information Model exists to compare against future needs. If no current Information Model exists, it needs to be created.
- Keep the 'to-be' processes in mind at all times. This will help avoid the tendency to revert to current, familiar processes (if they have changed), and recreate what is already in place.
- Evaluate the necessary reports to ensure all of the data elements are listed. It is considerably more expensive to add additional data elements later in the design phase.

No documents found

EXHIBIT III